How to merge a blipmovie with custom data

Here is a script which reads a blipmovie, overlays an op-art pattern,
and writes the result to a raw archive.

The script and source blipmovie are attached to the page below. Note
that the input
blipmovie contains a data hole at the centre.

```
## overlay.R: script to read a blipmovie, overlay some raw data, and
## write output to a rawarch.
##
## This script runs without the radR event loop, processing each frame
## explicitly.  This means, among other things, you can't step through
## it one frame at a time in the GUI to see what it's doing.  You can
## of course open the output raw arch using the GUI, and step through tha
## You can also save the raw arch as a blipmovie using the GUI.
##
## To run this script, start radR, then in the console window, do:
##
## source("overlay.R")


## get blipmovie reader port
from = BLIPMOVIE$get.ports()[[1]]

## configure the filename
config(from, filename="test2.bm")

## turn off rawarch compression, which on my machine links to a broken li
RAWARCH$compress = FALSE

## get rawarch writer port
to = RAWARCH$get.ports()[[2]]

## set output filename
config(to, filename="testout.raw.biglist")

## "start" both ports
start.up(from)
start.up(to)
```

```
## get the table of contents of the source; it's the side effects of this
## call that matter, not the return value

get.contents(from)

## get scan info for first frame; subsequent scan info is incremental
seek.scan(from, 1, 1)
get.scan.info(from)

## skip to the 20th scan in the input, to avoid null learning scans, e.g
seek.scan(from, 1, 20)

## start a run in the output
start.run(to)

## set up an extmat for holding scans
dat = extmat('my merged scan', type="short")

## process 100 scans, overlaying with a psychospiral
for(i in 1:100) {
    ## get scan header and data into si, dat
    si = get.scan.info(from)
    get.scan.data(from, dat)

    ## generate the pattern:

    pat = round(2048 + 1024 * outer(1:si$pulses, 1:si$samples.per.pulse,

    ## overlay it by taking the maximum sample value in each slot

    dat[] = pmax(dat[], pat)

    ## set output scaninfo; NOTE: we do this due to a bug in
    ## put.scan.rawarch whereby the header parameter is ignored, and
    ## the scan header is grabbed from the global RSS.  FIXME: rewrite
    ## radR from scratch.

    RSS$scan.info = si

    ## write to the raw archive
    put.scan(to, si, dat)

    ## show progress
    cat("Finished scan ", i, "\n")
}
```

```
end.run(to)
shut.down(to)

## the raw arch can now be read using the radR GUI
```